

VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
HO CHI MINH UNIVERSITY OF TECHNOLOGY
FACULTY OF APPLIED SCIENCE



BDC 2022 SPRING TRAINING
ASSIGNMENT 3 REPORT

**THU THẬP NỘI DUNG VÀ PHÂN TÍCH CẢM XÚC
TỪ CÁC COMMENT TIẾNG VIỆT**

Instructor: Hoàng Lê Hải Thanh
Performer: Nguyễn Đắc Hoàng Phú - 2010514
Võ Đức Minh -
Đặng Quang Vinh -
Hoàng Đức Nguyên -

Ho Chi Minh City, March/2022

Contents

1	INTRODUCE	2
1.1	Problem	2
1.2	Purpose	2
1.3	To do list	2
2	THEORETICAL BASIS	3
2.1	PhoBERT	3
2.2	Regex	3
2.3	Docker	3
2.4	FastAPI	4
3	PROCESSING STEP	6
3.1	Step 1: Preprocessing	6
3.2	Step 2: Embedding with phoBert	7
3.3	Step 3: Model prediction	8
4	CODE - PROCESS DATA	9
4.1	Fix lỗi gõ phím với telex và vni	9
4.2	Fix lỗi lặp từ (thường sẽ đi kèm luôn teencode)	9
4.3	Fix lỗi thêm dấu câu	9
4.4	Fix lỗi viết sai chính tả	10
4.5	Fix lỗi câu tiếng Việt và câu tiếng Anh	10
4.6	Fix lỗi teen code, icon	10
5	RESULT	11
5.1	SVM	11
5.2	LSTM	11
5.3	biLSTM	11
5.4	CNN	12
5.5	MLP	12
5.6	Kết luận về độ chính xác	12
6	DEPLOYMENT	13
6.1	Xây dựng các API	13
6.2	Viết dockerfile	13
6.3	Tạo image và push lên Dockerhub	14
6.4	Tải về project và deploy bằng Pycharm	15
7	SUMMARY	18

1 INTRODUCE

Đề tài: “Thu thập nội dung và phân tích cảm xúc từ các comment tiếng Việt”.

1.1 Problem

Từ bao đời nay, tiếng Việt là một ngôn ngữ rất khó để có thể phân tích cảm xúc cũng như xác định rõ mục đích câu nói vì ngữ pháp của ngôn ngữ chúng ta vô cùng phức tạp và đòi hỏi sự am hiểu cao mới có thể thông thạo được. Trong thời đại công nghệ số hiện nay, vấn đề phân tích cảm xúc câu nói đã được quan tâm rất nhiều để có thể ứng dụng trong các sản phẩm công nghệ nhằm lọc comment trong các mạng xã hội lớn, thu thập mong muốn người dùng,...

Nhằm thực hiện mục tiêu ấy, nhiều sản phẩm đã ra đời nhằm phân tích được các lời nói tiếng Việt. Hôm nay, nhóm đề xuất thực hiện mô hình phân tích dữ liệu câu nói đầu vào là tiếng Việt và cho ra output là mức độ biểu cảm của câu nói ấy.

1.2 Purpuse

- Phát triển từ ASG 1 và ASG 2, ASG3 của nhóm sẽ đi phân tích dữ liệu tiếng Việt thu thập được.
- Với model mới được tạo, ứng dụng dự kiến sẽ có thể nhận biết được cảm xúc của các câu nói, comment, tin nhắn, từ đó giúp ứng dụng nhiều vào cuộc sống.
- Ngoài các dạng dữ liệu thông thường, do tính không nhất quán của các câu nói, ngoài ra còn có teencode, icon lẫn với nhau trong dữ liệu nên model sẽ đảm nhận luôn chức năng nhận biết biểu cảm của các teencode, icon đó.
- Đối với các dữ liệu bị sai chính tả, thiếu dấu câu hoặc bị lỗi gõ tiếng Việt như Talex thì model cũng được thêm chức năng xử lý các lỗi trên để có thể đưa ra nhận xét chính xác nhất về câu nói.
- Bên cạnh đó, nhóm sẽ ứng dụng nhiều công nghệ mới như Docker, FastAPI nhằm có thể dễ dàng quản lý, triển khai ứng dụng và tạo lập giao diện tương tác dễ dàng với người dùng.
- Mục tiêu cuối cùng đạt được của sản phẩm là tạo ra một ứng dụng có khả năng nhận input đầu vào là nội dung câu nói, sau đó sẽ phân tích biểu cảm của input và thông báo cho người dùng

1.3 To do list

- Web scraping (crawl data).
- Xử lý dữ liệu (data cleaning).
- Build model xác định cảm xúc (dựa vào mô hình từ điển và thống kê).
- Tìm hiểu và sử dụng Docker để có thể dễ dàng triển khai, quản lý, chia sẻ ứng dụng
- Tìm hiểu FastAPI để triển khai ứng dụng.

Hướng đi xa hơn qua kinh nghiệm từ project:

- Crawl các file dữ liệu khác (âm thanh (.wav), hình ảnh,...)
- Xây dựng datalake.

Đường dẫn homework: <https://github.com/metacrektalHCMUT/BDC-Assignment>

Đường dẫn landing page: <https://metacrektalhcmut.github.io/BDC-Assignment/>

2 THEORETICAL BASIS

2.1 PhoBERT

PHOBERT-EMBEDDING

Vấn đề đặt ra

Khi nhận vào dữ liệu là comment Tiếng Việt và bài toán đặt ra là cảm xúc của comment đó. Cảm xúc ở đây hoàn toàn có thể là tích cực, xấu đi, trung tính hoặc đơn thuần là Tốt / Xấu.

Lớp bài toán này thì những làm là tất cả chúng ta sẽ vector hoá câu văn bản, sau đó đưa vào một mạng LSTM để lấy ra vector output.

PhoBERT là gì?

PhoBERT là một pretrain model sử dụng mô hình attention network sử dụng cho các tác vụ liên quan đến tiếng Việt. Nó là một module của thư viện transformer hỗ trợ về việc trích xuất các văn bản tiếng Việt thành một lớp embedding với 256 feature.

PhoBERT được train trên khoảng 20GB dữ liệu bao gồm khoảng 1GB Vietnamese Wikipedia corpus và 19GB còn lại lấy từ Vietnamese news corpus. Đây là một lượng dữ liệu khá ổn để train một mô hình như BERT.

2.2 Regex

SỬA LỖI CHÍNH TẢ VÀ EMOJI VỚI REGEX

Vấn đề đặt ra

Khi nhận vào comment Tiếng Việt có lỗi chính tả hoặc có emoji, bài toán đặt ra ở đây là công cụ để sửa lỗi chính tả và chuyển đổi emoji thành ý nghĩa tương ứng.

Regex là gì?

Regex hay biểu thức chính quy (Regular Expression) là một chuỗi ký tự tạo thành một biểu mẫu tìm kiếm (search pattern). Regex được sử dụng để kiểm tra xem một chuỗi có chứa mẫu tìm kiếm được chỉ định hay không.

Các hàm xử lý Regex

findall	Trả về một list các kết quả phù hợp
search	Trả về một Match object nếu có bất kỳ vị trí nào trong chuỗi phù hợp
split	Trả về một list chuỗi đã được phân chia ở vị trí match
sub	Thay thế các vị trí match với biểu thức bằng một chuỗi khác

2.3 Docker

QUẢN LÝ, CHIA SẺ VÀ DEPLOY ỨNG DỤNG VỚI DOCKER

Vấn đề đặt ra

Việc setup và deploy application lên một hoặc nhiều server rất vất vả từ việc phải cài đặt các công cụ, môi trường cần cho application đến việc chạy được ứng dụng chưa kể việc không đồng nhất giữa các môi trường trên nhiều server khác nhau. Đồng thời bên cạnh đó, việc setup môi trường sẽ mất rất nhiều thời gian và đôi khi xảy ra lỗi. Chính vì lý do đó Docker được ra đời để giải quyết vấn đề này.

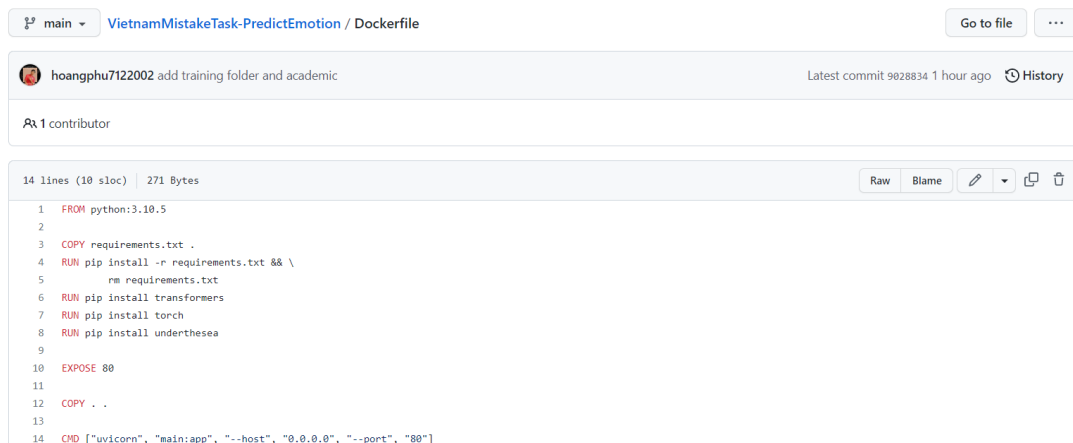
Docker là gì?

Docker là một nền tảng cho developers và sysadmin để develop, deploy và run application với container. Nó cho phép tạo các môi trường độc lập và tách biệt để khởi chạy và phát triển ứng dụng và môi trường này được gọi là container. Container của docker sẽ gom tất cả các công cụ, môi trường, source code của developer lại thành một và dễ dàng vận chuyển, chia sẻ. Khi cần deploy lên bất kỳ server nào chỉ cần run container của Docker thì application của bạn sẽ được khởi chạy ngay lập tức. **Một số thuật ngữ về Docker:**

- Docker file: là file dùng để hướng dẫn cho hệ thống cách build các image.
- Image: là dẫn xuất từ docker file, được tạo thành sau khi build docker file. Đây là nơi chứa hệ điều hành, các công cụ, thư viện, source code tương ứng của chương trình tương thích với chương trình nguyên mẫu được viết bởi developer.
- Container: là dẫn xuất của image, khi run image, container sẽ được tạo, tương ứng với chương trình sẽ được chạy.
- Docker Hub: là Registry lớn nhất của Docker Images (mặc định). Có thể tìm thấy images và lưu trữ images của riêng bạn trên Docker Hub (miễn phí).

Trong project lần này, nhóm sẽ dùng docker để có thể dễ dàng lưu trữ và vận chuyển, chia sẻ ứng dụng cho các thành viên cũng như dễ dàng triển khai cho người dùng.

Dưới đây là hình ảnh docker file của nhóm, từ docker file ta có thể chạy ra các image, rồi từ image chạy ra các container để deploy ứng dụng dễ dàng trên các sever khác nhau mà không cần lo conflict hay thiếu thư viện.



```
1 FROM python:3.10.5
2
3 COPY requirements.txt .
4 RUN pip install -r requirements.txt && \
5     rm requirements.txt
6 RUN pip install transformers
7 RUN pip install torch
8 RUN pip install underthesea
9
10 EXPOSE 80
11
12 COPY . .
13
14 CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "80"]
```

Sử dụng docker trong project

2.4 FastAPI

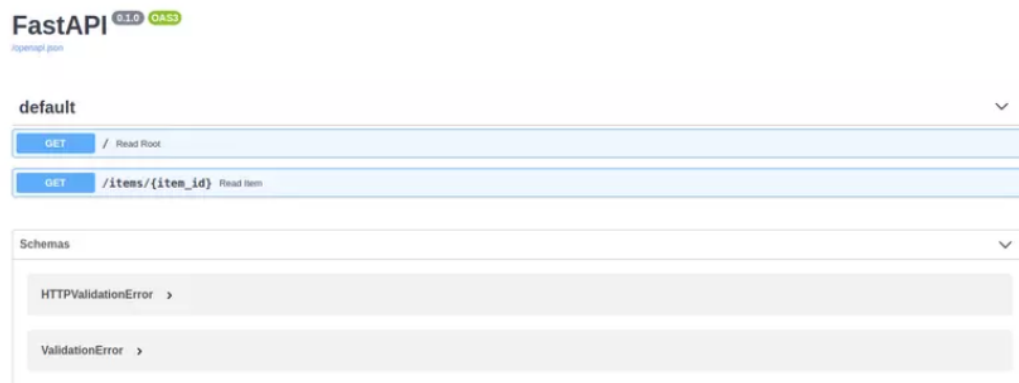
SỬ DỤNG ỨNG DỤNG VỚI FASTAPI

Vấn đề đặt ra?

Sau khi project được hoàn thiện, model được tạo ra có khả năng thực hiện nhiều chức năng khác nhau như sửa teencode, thêm accent, đánh giá biểu cảm câu nói, ... Tuy nhiên các model trên chỉ đơn giản là code và khó có thể cho người dùng dễ dàng sử dụng vào mục đích cụ thể.

FastAPI là gì?

FastAPI là một framework dùng để tạo API cho người dùng có thể tương tác được với model của nhóm. Do based trên OpenAI mà trước đó có tên là Swagger nên FastAPI cung cấp doc có giao diện dễ nhìn, dễ sử dụng. Khi bật doc bằng local url <http://0.0.0.0:8000/docs>:



Giao diện tương tác của FastAPI

Với mỗi model, ta sẽ tạo ra một api tương ứng để người dùng có thể tương tác. Ta sẽ sử dụng FastAPI để tạo các luồng truy cập tới từng model với chức năng khác nhau, luồng sẽ do ta đặt tên

```
16 class Request(BaseModel):
17     text: str
18
19 @app.post("/correctTeencode")
20 def teencode(data: Request):
21     data = data.dict()
22     corrected = correct_sent(data["text"])
23     return {"result": corrected}
24
25 @app.post("/accent")
26 def accent(data: Request):
27     data = data.dict()
28     corrected = returnRealOutput(data["text"], model)
29     return {"result": corrected}
30
31 @app.post("/emotion")
32 def emotion(data: Request):
33     data = data.dict()
34     corrected = returnRealOutput(correct_sent(data["text"]), model)
35     return {"result": predict(corrected)}
36
```

Sử dụng FastAPI để người dùng dễ dàng tương tác với model

Khi người dùng muốn sử dụng model để sửa teencode, chỉ cần truy cập vào link <http://0.0.0.0:8000/docs/correctTeencode> sau đó đưa dữ liệu cần correct vào thì sẽ tương tác được với model và nhận được kết quả như mong muốn. Ta thực hiện tương tự đối với các chức năng khác.

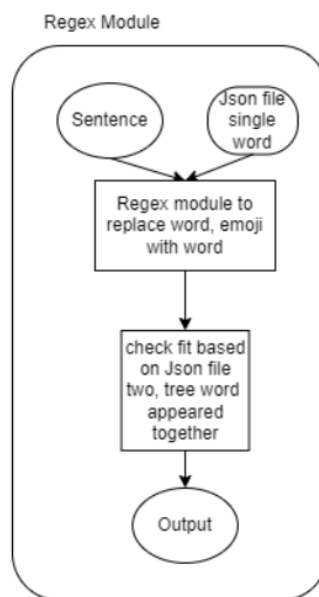
3 PROCESSING STEP

Bao gồm 3 bước, được thể hiện tổng quát dưới đây.

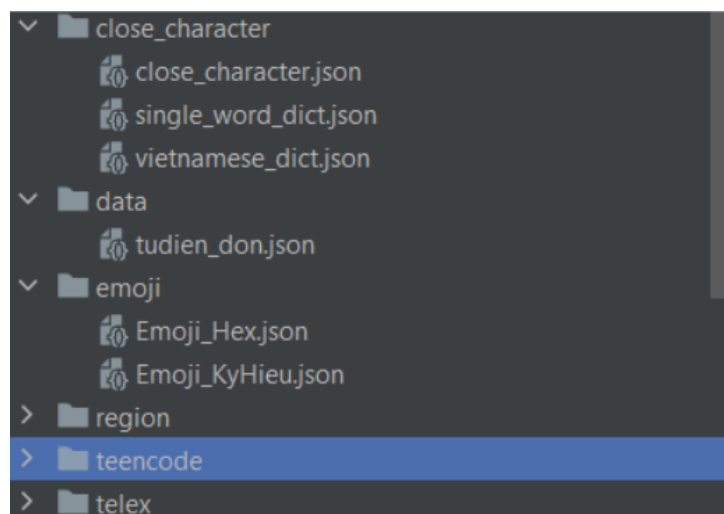
Từ comment ban đầu, qua preprocess bao gồm fix lỗi chính tả, thêm dấu,... sẽ đưa ra một clean text, từ clean text qua API của phoBert để embedding ra vector.

3.1 Step 1: Preprocessing

Regex module



- Từ comment ban đầu, qua preprocess, đầu vào sẽ là câu cần correct và các file json chứa các từ đơn để thay thế.
- Kiểm tra các từ đơn đã thay thế có phù hợp với các từ phía xung quanh, nếu phù hợp thì giữ lại.
- Những tác vụ: fix lỗi phím gần với vni, telex, lỗi phát âm vùng miền, địa phương, lỗi teencode và sử dụng emoji.



.JSON: Các từ điển sử dụng cho tác vụ

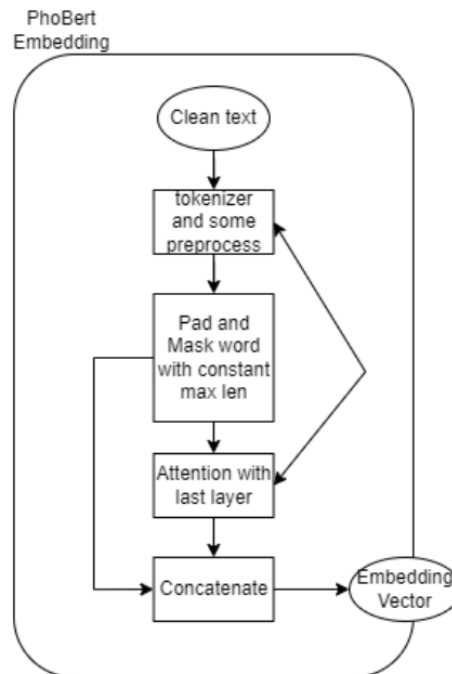
```
def correct_sent(sent):
    sent = fix_emoji(sent)
    sent = preprocess(sent)
    sent = correct_short_word_sent(sent)
    sent = use_correct_func(sent, telexCorrector.fix_telex_word)
    sent = use_correct_func(sent, vniCorrector.fix_vni_sentence)
    sent = use_correct_func(sent, correct_teencode_word)
    sent = use_correct_func(sent, vniCorrector.fix_vni_sentence)
    sent = use_correct_func(sent, telexCorrector.fix_telex_word)

    sent = correct_close_character_sent(sent)
    sent = southNorthProcess(sent)

    return sent
```

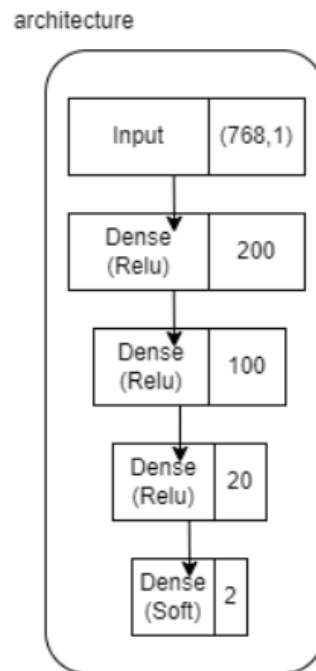
CORRECT_SENT: STEP-BY-STEP sửa lỗi chính tả

3.2 Step 2: Embedding with phoBert



- Từ clean text, ta tiến hành tokenize với thư viện underthesea, sau đó padding câu đến chung độ dài MAX
- Với các câu như vậy, ta đánh dấu điểm nào là được thêm vô qua bước lọc Mask, rồi đưa qua layer attention cuối cùng của PhoBert Embedding
- Sau đó với câu đã predict được, qua một từ điển chứa các từ có nghĩa để xem xét chữ đó có nên được thay thế hay không

3.3 Step 3: Model prediction



- Dense layer là một lớp cổ điển trong Neural network. Mỗi nơ ron nhận đầu vào từ tất cả nơ ron lớp trước đó.
- ReLU (Rectified Linear Unit) là một thành phần của một tế bào thần kinh nhân tạo trong các mạng thần kinh nhân tạo (ANN), chức năng kích hoạt là trách nhiệm xử lý đầu vào trọng và giúp đỡ để cung cấp một sản lượng.
- Softmax hàm nhận vào một vectơ của chứa K số thực và chuẩn hóa nó thành phân phối xác suất chứa K xác suất tỷ lệ thuận với lũy thừa của các số đầu vào.

4 CODE - PROCESS DATA

4.1 Fix lỗi gõ phím với telex và vni

Vấn đề đặt ra:

- Khi gõ phím để hoàn thiện câu nói, lời văn trên các thiết bị máy tính, người dùng hay gặp phải lỗi gõ phím tiếng Việt với các bộ Font như Talex hay Vini.
- Ví dụ: "á" (từ muốn viết) - "as" (Talex) - "a1" (Vini).
- Khi thu thập dữ liệu, chúng ta sẽ không thể nào loại trừ các trường hợp trên, chính vì thế cần xây dựng tính năng xử lý lỗi gõ phím tiếng Việt.

Cách giải quyết

- Xem ngữ cảnh xung quanh: có chữ lớp, biển báo, cách, hoặc có hay không chữ đứng trước ký tự VD: ha3 hoặc a3).
- Dùng regex để bắt, chú ý thêm những trường hợp có dấu cách quá xa kiểu: "a 3" (dễ nhầm lẫn với a 3/2/2022).
- Thường các cặp từ này đi với nhau không có nghĩa như ax, ar, aw, giải quyết thêm các trường hợp có 2 chữ trở lên vì phạm teen code.

4.2 Fix lỗi lặp từ (thường sẽ đi kèm luôn teencode)

Vấn đề đặt ra:

- Khi gõ phím để hoàn thiện câu nói, lời văn trên các thiết bị máy tính, người dùng hay gặp phải tình trạng gõ lặp một chữ cái khiến cho từ không được hình thành một cách hoàn chỉnh.
- Ví dụ: chooooojn => chọn, ddos => đó.
- Khi thu thập dữ liệu, chúng ta cần sửa các từ lại cho đúng chính tả.

Cách giải quyết

- Thường các chữ này nhân lên, tạo 26 pattern cho các từ có 2 chữ trở lên đi kèm với nhau, thay nó bằng 1 chữ.
- Chú ý chữ ee => ê, chữ dd thành chữ đ, chữ oo thành chữ ô.

4.3 Fix lỗi thêm dấu câu

Vấn đề đặt ra:

- Vì một số nguyên nhân khách quan như: muốn tạo ấn tượng, máy tính chưa cài font Việt hóa, lỗi font tiếng Việt mà nhiều trường hợp câu nói của người dùng bị thiếu mất dấu câu.
- Ví dụ: neu => nếu, dau => đau or dầu.
- Khi thu thập dữ liệu, chúng ta cần thêm dấu câu để thu nhận kết quả chính xác nhất.

Cách giải quyết

- Ưu tiên character level hơn word level.
- Dùng các model như LSTM (với fixed ngram), Transformer (ưu tiên) để thử nghiệm.

4.4 Fix lỗi viết sai chính tả

Vấn đề đặt ra:

- Do hiểu biết của người dùng sai về mặt từ ngữ hoặc do muốn tạo điểm nhấn cho câu nói, một số dữ liệu thu thập được sẽ có các từ ngữ sai chính tả.
- Ví dụ: cúc áo => cúc áo, nhấn nút => nhấn nút.
- Khi thu thập dữ liệu, chúng ta cần sửa lại chính tả của từng từ ngữ để thu nhận kết quả chính xác nhất.

Cách giải quyết

- Tập trung kiểm tra các từ ghép, từ láy thường gặp.
- So sánh các từ với từ điển.

4.5 Fix lỗi câu tiếng Việt và câu tiếng Anh

Vấn đề đặt ra:

- Có những câu nói mà người Việt lại dùng quá nhiều từ ngữ tiếng Anh trong câu dẫn tới khi đưa dữ liệu vào sẽ không thể thực hiện các chức năng của chương trình.
- Ví dụ: "Bạn sống healthy và balance quá! ".
- Cần nhận định đúng câu nói là tiếng Anh hay tiếng Việt trước khi đưa vào model.

Cách giải quyết

- Có một tập từ điển Anh Việt lớn và tốt.
- Dựa vào ngữ cảnh xem xét, thường là địa chỉ, tên người, tên đường, tên phim, tên chương trình (nói chung là danh từ).

4.6 Fix lỗi teen code, icon

Vấn đề đặt ra:

- Với sự phát triển của ngôn ngữ hiện đại, giới trẻ dần sử dụng teen code cũng như chèn thêm các icon trong câu nói để rút gọn văn phong cũng như tạo ấn tượng mạnh mẽ với đối tượng giao tiếp.
- Ví dụ: ik => đi, j => gì, og => ông, icon (:>, :))) . :((,....)
- Cần đưa teen code về ngôn ngữ thuần túy cũng như dịch nghĩa các icon trong câu nói trước khi đưa vào model.

Cách giải quyết

- Dựa vào từ điển bỏ đi các thành phần dư.
- Sau đó sử dụng model thêm dấu.
- Đối với những câu thuần teencode thì cần có một từ điển riêng để map.

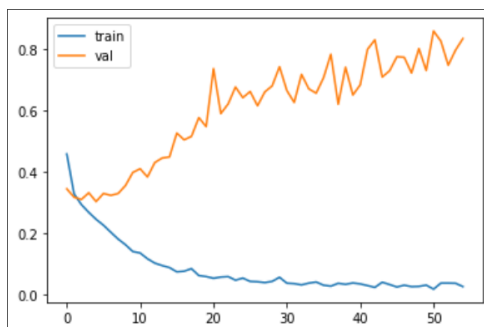
5 RESULT

5.1 SVM

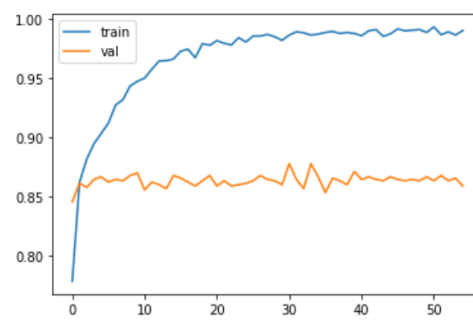
Đây là mô hình ML về việc với một vector mới, sẽ chọn các vector có khoảng cách gần nhất và dựa vào đó để rút trích đặc trưng đưa ra kết quả dự đoán.

5.2 LSTM

Đây là mô hình chuỗi sử dụng cho các data dạng seq2seq, việc học đặc trưng mang tính chất toàn cục



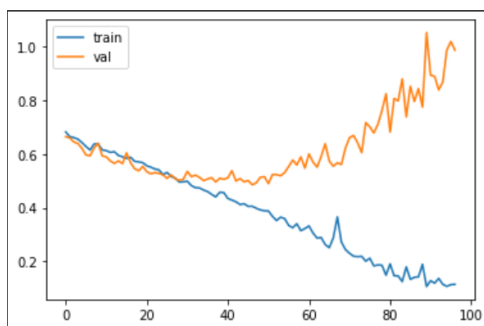
(a) *loss*



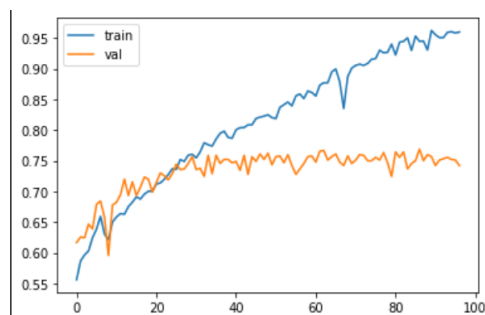
(b) *accuracy*

5.3 biLSTM

Là mô hình cải tiến của LSTM với việc dùng các node ở tương lai để điều chỉnh sai lệch các tính toán trong quá khứ.



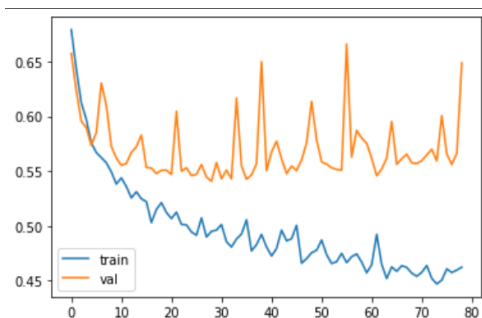
(a) *loss*



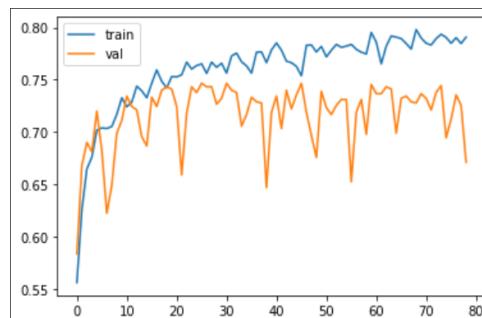
(b) *accuracy*

5.4 CNN

Đây là mô hình thường sử dụng trong các tác vụ liên quan đến việc xử lý hình ảnh. Nhưng vẫn có thể sử dụng cho các bài toán văn bản với việc học đặc trưng dựa vào tính cục bộ,



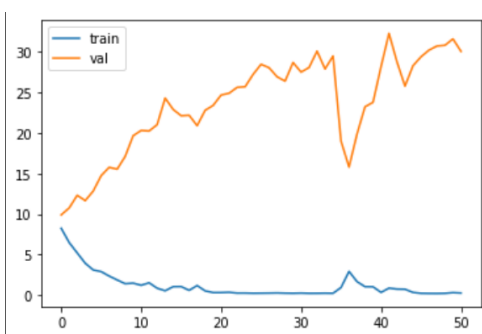
(a) *loss*



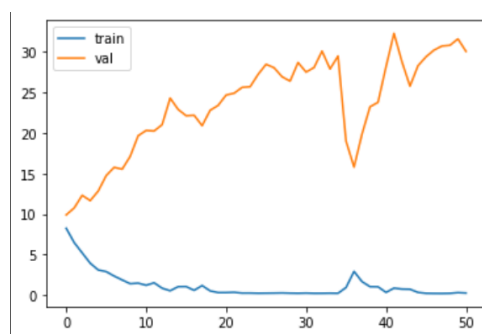
(b) *accuracy*

5.5 MLP

Là một mô hình neural network với việc chồng các layer lên nhau, mỗi layer sẽ liên quan tới việc trích xuất đặc trưng khác nhau.



(a) *loss*



(b) *accuracy*

5.6 Kết luận về độ chính xác

MODEL	loss	accuracy
SVM	NULL	0.876
LSTM	0.3021	0.8740
CNN	0.5620	0.7040
MLP	0.0936	0.8900
biLSTM	0.5053	0.7480

Do model MLP có độ chính xác cao nhất cũng như độ mất mát thấp nhất nên ta sử dụng model này để dự đoán các dữ liệu Tiếng Việt từ các trang web.

6 DEPLOYMENT

6.1 Xây dựng các API

Để có thể triển khai model với FastAPI thì điều đầu tiên ta cần làm là xây dựng các API cho từng phương thức

```
8
9  app = FastAPI()
10 model = Model()
11
12 @app.get("/")
13 def home():
14     return "Congratulations! Your API is working as expected. This new version allows for batching. Now head over to http://localhost:81/
15
16 class Request(BaseModel):
17     text: str
18
19 @app.post("/correctTeencode")
20 def teencode(data: Request):
21     data = data.dict()
22     corrected = correct_sent(data["text"])
23     return {"result": corrected}
24
25 @app.post("/accent")
26 def accent(data: Request):
27     data = data.dict()
28     corrected = returnRealOutput(data["text"], model)
29     return {"result": corrected}
```

Bên trong các đường dẫn ta xây dựng các hàm phù hợp IO với từng phương thức POST, GET. Có thể xây dựng API theo dạng Batch hoặc Request tùy mục đích sử dụng.

6.2 Viết dockerfile

```
1 FROM python:3.10.5
2
3 COPY requirements.txt .
4 RUN pip install -r requirements.txt && \
5     rm requirements.txt
6 RUN pip install transformers
7 RUN pip install torch
8 RUN pip install underthesea
9
10 EXPOSE 80
11
12 COPY . .
13
14 CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "80"]
```

Để có thể triển khai bằng Docker, ta cần viết 1 file Dockerfile chứa tất cả các môi trường cần có để có thể khởi chạy ứng dụng. Bên cạnh đó, trong Dockerfile ta cũng sẽ chọn port để khởi chạy API trên localhost và copy các tệp cần chạy của ứng dụng và các lệnh cần thiết để khởi chạy ứng dụng sau khi việc thiết lập hoàn thành.

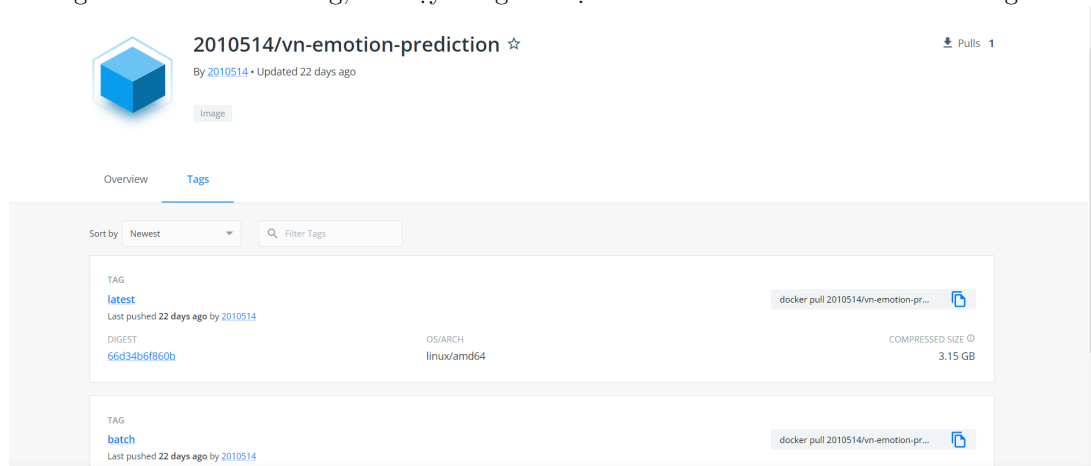
6.3 Tạo image và push lên Dockerhub

```
[+] Building 5.5s (13/13) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.10.5
=> [auth] library/python:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 8.00MB
=> [1/7] FROM docker.io/library/python:3.10.5@sha256:feebf94b56ce1f20a29427ff697bccbc659952f0
=> CACHED [2/7] COPY requirements.txt .
```

Từ Dockerfile, ta tiến hành build image với command line: `docker build -t 2010514:test .`

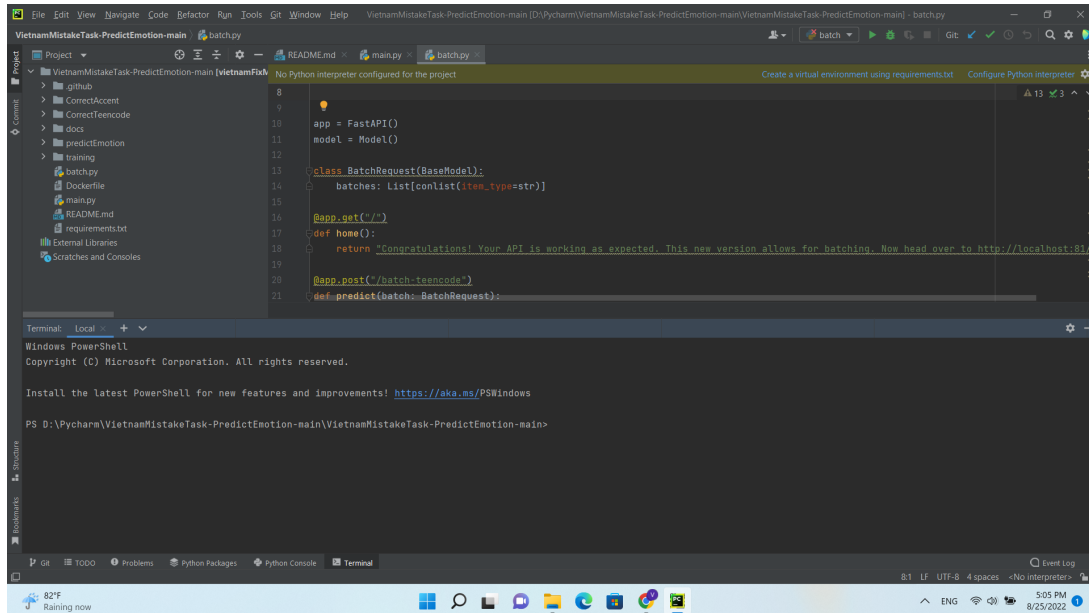
```
[+] Building 5.5s (13/13) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.10.5
=> [auth] library/python:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 8.00MB
=> [1/7] FROM docker.io/library/python:3.10.5@sha256:feebf94b56ce1f20a29427ff697bccbc659952f0
=> CACHED [2/7] COPY requirements.txt .
```

Với image đã build thành công, ta chạy image để tạo thành container để làm môi trường cho API



Sau khi build thành công, ta tiến hành push container lên dockerhub.

6.4 Tải về project và deploy bằng Pycharm



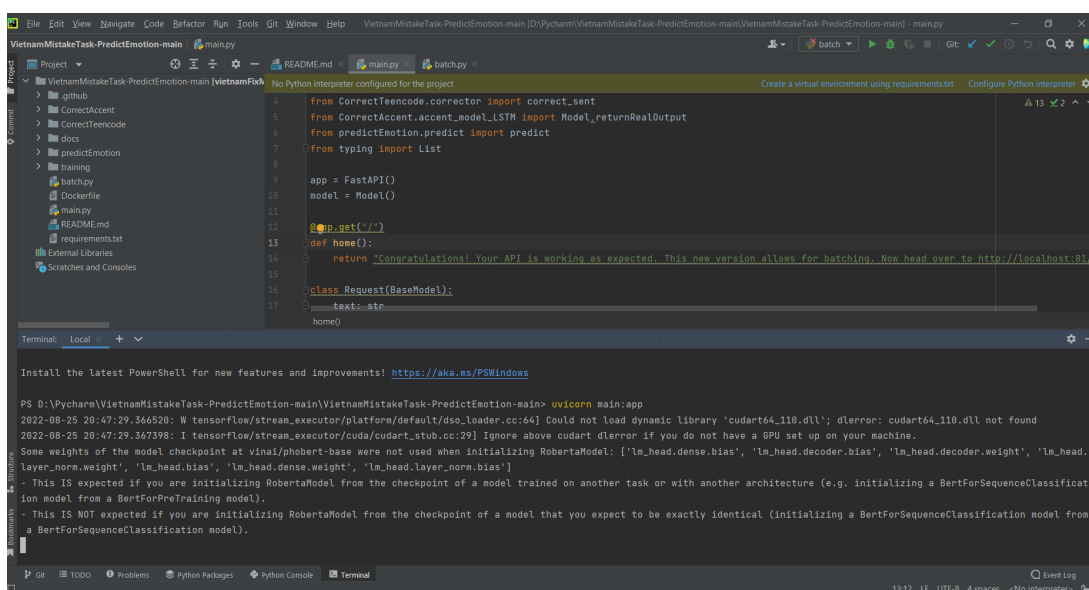
Đây là giao diện khi mới tải về của project

Để khởi chạy dự án và tương tác với model, ta sẽ cần khởi động file main.py và tạo ra một sever gồm các API giúp người dùng có thể tương tác với các model.

Các bước thực hiện

1. Chạy dòng lệnh `uvicorn main:app` để khởi chạy file main.py. Lúc này, chương trình sẽ trả về một API trang web local tạo ra giao diện giúp người dùng tương tác với model.
2. Mở API được trả về và thêm giao thức `docs` phía sau API. Lúc này, giao diện chính sẽ được mở ra.
3. Chọn model muốn tương tác, chọn lệnh **Try it out** và thay string trong file json bằng dòng ký tự muốn đưa vào model, sau đó chọn execute và nhận kết quả

Minh họa các bước thực hiện



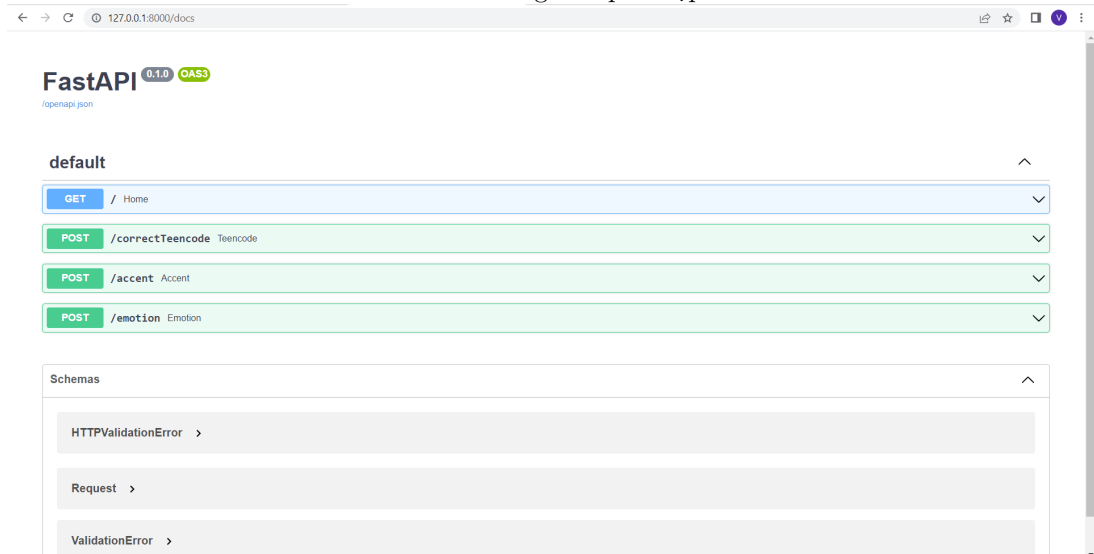
Chạy file main.py


```
2022-08-25 20:47:55.986542: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cusparses64.11.dll'; dlerror: cusparses64.11.dll not found
2022-08-25 20:47:55.986897: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudnn64.8.dll'; dlerror: cudnn64.8.dll not found
2022-08-25 20:47:55.987044: W tensorflow/core/common_runtime/gpu/gpu_device.cc:1850] Cannot dlopen some GPU libraries. Please make sure the missing libraries mentioned above are installed
properly if you would like to use GPU. Follow the guide at https://www.tensorflow.org/install/gpu for how to download and setup the required libraries for your platform.
Skipping registering GPU devices...
2022-08-25 20:47:55.914219: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following
CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
INFO: Started server process [16344]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
```

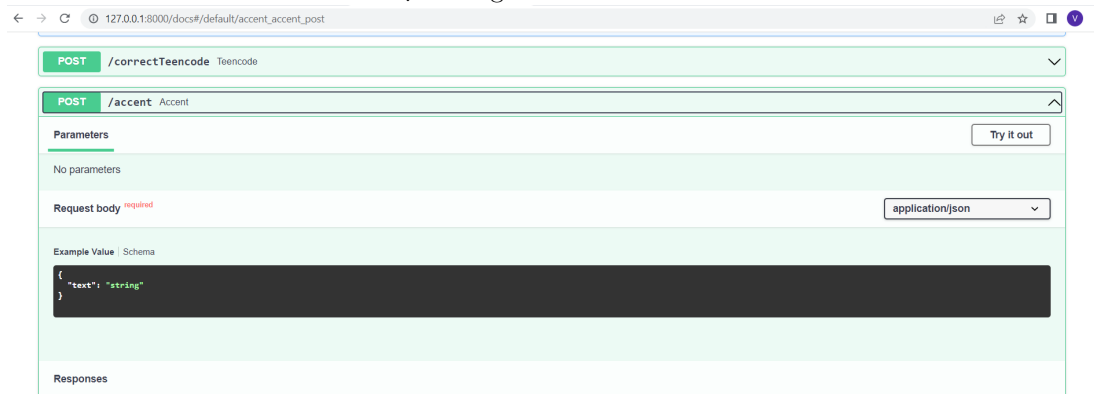
Nhận API được trả về



Thêm đường dẫn phù hợp



Giao diện tương tác chính với các model



Chọn model muốn tương tác và chọn try it out để tương tác với model

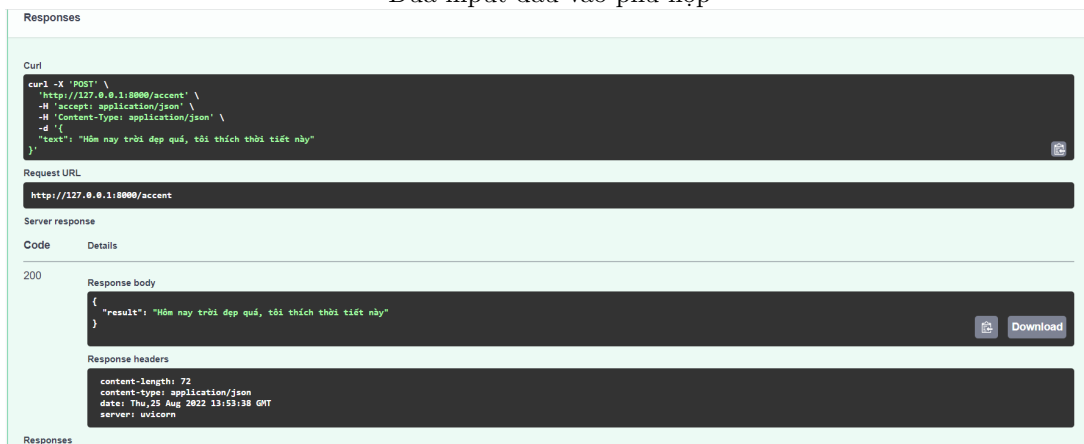


The image shows a web-based API client interface. At the top, there's a 'Parameters' tab with 'Cancel' and 'Reset' buttons. Below it, a message says 'No parameters'. The 'Request body' section is marked as 'required' and has a dropdown menu set to 'application/json'. A text area contains a JSON object:

```
{  "text": "Hôm nay trời đẹp quá, tôi thích thời tiết này"}
```

. At the bottom of the request section are two buttons: 'Execute' (highlighted in blue) and 'Clear'. Below the request section is a 'Responses' section, which is currently empty.

Đưa input đầu vào phù hợp



The image shows the same API client interface after the request has been executed. The 'Responses' section is now populated. It shows the 'Curl' command used for the request, the 'Request URL' as 'http://127.0.0.1:8000/accnt', and the 'Server response' with a status code of 200. The 'Response body' is a JSON object:

```
{  "result": "Hôm nay trời đẹp quá, tôi thích thời tiết này"}
```

. The 'Response headers' are also displayed:

```
content-length: 72
content-type: application/json
date: Thu, 25 Aug 2022 13:53:38 GMT
server: uvicorn
```

. There is a 'Download' button next to the response body.

Chọn Excecute và nhận kết quả trả về

7 SUMMARY

Trong assignment 3, nhóm tập trung vào các vấn đề chính, bao gồm

- Xử lý dữ liệu đầu vào, thu thập và xử lý các comment tiếng Việt, từ các câu dài chuyển thành các vector dùng cho các bước tiếp theo.
- Đánh giá hiệu năng của model được train từ những assignment trước, thông qua đó rút được những thông số cần thiết để cải thiện model.
- Sau các công đoạn trên, đóng gói sản phẩm thành api thông qua FastAPI và Docker với tiêu chí nhanh chóng, thuận tiện.